
A Ruby meta-golf session
PHRUG
Philippine Ruby Users Group

The Challenge

https://code.golf/gijswijts-sequence#ruby

The Starter Code

https://gitlab.com/-/snippets/4808279

PHRUG
Philippine Ruby Users Group

Step 1

Iterate 1000 times over an array that:

- Puts the last item
- Performs the required evaluation logic
- appends the result to the array

Results in 1000 entries of result '1'

```ruby
g=[1]

1000.times do |i|
p g[-1]
    s=1
    g.append(s)
end
```

Output

```
1
1
1
1
1
1
1
1
1
```

Diff

	Expected	Output
1	1	1
2	1	1
3	2	
4	1	1
5	1	1
6	2	

PHRUG
Philippine Ruby Users Group

Step 2

Add a line to dump your analysis on any chosen line

```ruby
g=[1]
ln = 3

1000.times do |i|
p g[-1]
    s=1
    print "gijswits dump: " if i == ln
    g.append(s)
end
```

PHRUG
Philippine Ruby Users Group

Step 3

For line 3, need to compare arr[-1] and arr[-2]
Use a 'stabby' lambda

```
[1]
[1],[1]
 1 , 1 ,[2]
 1 , 1 , 2 ,[1]
 1 , 1 , 2 ,[1],[1]
[1 , 1 , 2],[1 , 1 , 2]
 1 , 1 , 2 , 1 , 1 ,[2],[2]
 1 , 1 , 2 , 1 , 1 ,[2],[2],[2]
 1 , 1 , 2 , 1 , 1 , 2 , 2 , 2 ,[3]
 1 , 1 , 2 , 1 , 1 , 2 , 2 , 2 , 3 , 1
```

```ruby
arr=[1]
lam = -> (p1, p2) { p1 == p2 }
line = 1

1000.times do |i|
    p arr[-1]
    s=1
    print "arr[-1]: #{arr[-1]}, \
    arr[-2]: #{arr[-2]}, \
    lam.call(arr[-1],arr[-2]): \
    #{lam.call(arr[-1],arr[-2])} " if i == line

    arr.append(s)
end
```

Output

```
1
1
arr[-1]: 1,     arr[-2]: 1,     lam.call(arr[-1],arr[-2]):      true 1
1
1
1
1
1
```

Diff

	Expected		Output
1	1	1	1
2	1	2	1
3	2	3	arr[-1]: 1, arr[-2]: 1, lam.call(arr[-1],arr[-2]): true 1
4	1	4	1
5	1	5	1

PHRUG
Philippine Ruby Users Group

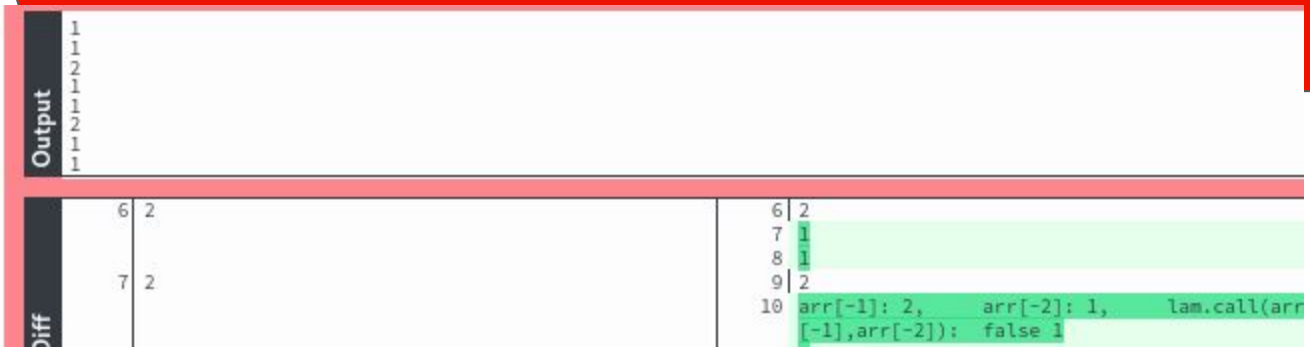
Step 4

Increment ‘s’ using the lambda

Move line_dump several lines to see next failure

```
arr=[1]
lam = -> (p1, p2) { p1 == p2 }
dump_line = 8

1000.times do |i|
    p arr[-1]
    s=1
    print "arr[-1]: #{arr[-1]}, \
    arr[-2]: #{arr[-2]}, \
    lam.call(arr[-1],arr[-2]): \
    #{lam.call(arr[-1],arr[-2])} " if i==dump_line
    s+=1 if lam.call(arr[-1],arr[-2])
    arr.append(s)
end
```



PHRUG
Philippine Ruby Users Group

Step 5

Extract dump code to proc to save space

Note the change to calling the Proc with '[ ]'

```
arr=[1]
lam = -> (p1, p2) { p1 == p2 }
dump_line = 5
gijs_dump = ->(s1, s2, lam) {
    print "s1: #{s1}, \
    s2: #{s2}, \
    lam[s1,s2]: #{lam[s1,s2]} "
}

1000.times do |i|
    p arr[-1]
    s=1
     gijs_dump[arr[-1], arr[-2], lam] if i==dump_line
    s+=1 if lam[arr[-1],arr[-2]]
    arr.append(s)
end
```

Will be 1-line in future steps

```
Output
1
1
2
1
1
2
s1: 2,  s2: 1,  lam[s1,s2]: false 1
1
```

```
Diff
6 2                          6 2
                             7 s1: 2,  s2: 1,  lam[s1,s2]: false 1
                             8 1
7 2                          9 2
                             10 1
                             11 1
8 2                          12 2
```

PHRUG
Philippine Ruby Users Group

Step 6

For line 6, need to compare arr[-3..-1] with arr[-4..-6]

Without breaking earlier logic

```
[1]
[1],[1]
 1 , 1 ,[2]
 1 , 1 , 2 ,[1]
 1 , 1 , 2 ,[1],[1]
[1 , 1 , 2],[1 , 1 , 2]
 1 , 1 , 2 , 1 , 1 ,[2],[2]
 1 , 1 , 2 , 1 , 1 ,[2],[2],[2]
 1 , 1 , 2 , 1 , 1 , 2 , 2 , 2 ,[3]
 1 , 1 , 2 , 1 , 1 , 2 , 2 , 2 , 3 , 1
```

```ruby
arr=[1]
lam = -> (p1, p2) { p1 == p2 }
dump_line = 8
gijs_dump = ->(s1, s2, lam) { print "s1: #{s1}, s

10.times do |i|
  p arr[-1]
  s=1
  (0..i).each do |f|
    gijs_dump[arr[-1-f..], arr[-2-2*f..-2-f], lam
    if lam[arr[-1-f..],arr[-2-2*f..-2-f]]
      s+=1
      break
    end
  end
  arr.append(s)
end
```

Output

```
2
1
1
2
2
2
s1: [2], s2: [2], lam[s1,s2]: true 2
2
```

Diff

```
 7 | 2                                  | 7 | 2
 8 | 2                                  | 8 | 2
 9 | 3                                  | 9 | s1: [2], s2: [2], lam[s1,s2]: true 2
10 | 1                                  |   |
11 | 1                                  |   |
12 | 2                                  |10 | 2
13 | 1                                  |   |
```

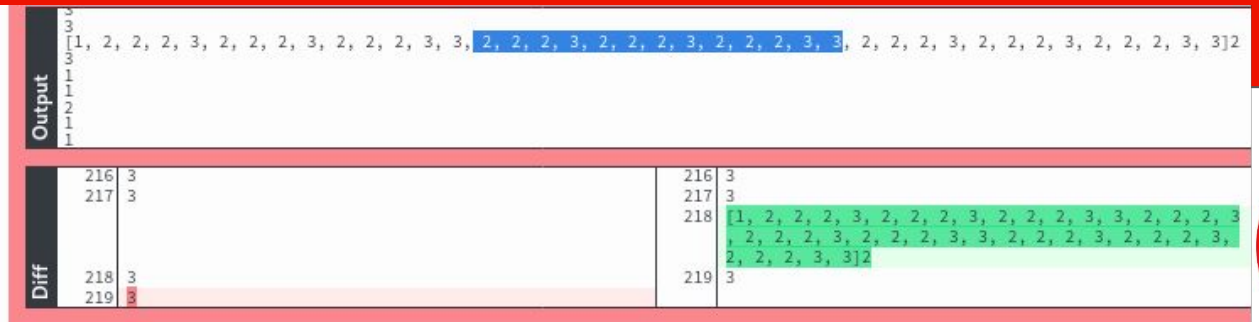
PHRUG
Philippine Ruby Users Group

Step 8

For line 8, need to compare arr[-1] with arr[-2] with arr[-3]

Without breaking earlier logic

```ruby
1000.times do |i|
  p arr[-1]
  s=1
  (0..i).each do |f|
    # gijs_dump[arr[-3-3*f..-3-2*f],arr[-4-4*f..-4-3*f], ]
    if lam[arr[-1-f..],arr[-2-2*f..-2-f]]
      s+=1
      if lam[arr[-2-2*f..-2-f],arr[-3-3*f..-3-2*f]]
        s+=1
      end
    break
    end
  end
  arr.append(s)
end
```



PHRUG
Philippine Ruby Users Group

Step 9

Add extra print statements to better understand this error

‘break’ statement exits loop early during small repetitions

In this case: [3,3]

```
f1='arr[-1-f..]'
f2='arr[-2-2*f..-2-f]'
f3='arr[-3-3*f..-3-2*f]'
f4='arr[-4-4*f..-4-3*f]'

1000.times do |i|
  p arr[-1]
  s=1
  print arr[-40..] if i==dump_line
    (0..i).each do |f|
        gijs_dump[eval(f1),eval(f2), lam] if i==dump_line
        if lam[eval(f1),eval(f2)]
            s+=1
            gijs_dump[eval(f2),eval(f3), lam] if i==dump_line
            if lam[eval(f2),eval(f3)]
                gijs_dump[eval(f3),eval(f4), lam] if i==dump_line
                s+=1
            end
        break
```

Extract to strings to reduce duplication

Execute using ‘eval’

‘break’ causes early exit

Output

```
2, 2, 3, 3, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2, 2, 3, 3]s1: [3], s2: [3], lam[s1,s2]: true s1: [3], s2: [2], lam[s1,s2]: fals
```

Diff

```
218 3
219 3
220 4
221 1

, 2, 2, 2, 3, 2, 2, 2, 3, 3, 2, 2, 2, 3, 2, 2, 2, 3
2, 2, 2, 3, 3]s1: [3], s2: [3], lam[s1,s2]: true s1
[3], s2: [2], lam[s1,s2]: false 2
219 3
220 1
```

PHRUG
Philippine Ruby Users Group

Step 10

Add magic numbers to resolve this later

```ruby
1000.times do |i|
    p arr[-1]
    s=1
    # print arr[-40..] if i==dump_line
    (0..i).each do |f|
        s+=1 if [216,436,657,877].include?(i)
        gijs_dump[eval(f1),eval(f2), lam] if i==dump_line
        if lam[eval(f1),eval(f2)]
            s+=1
            gijs_dump[eval(f2),eval(f3), lam] if i==dump_line
            if lam[eval(f2),eval(f3)]
                gijs_dump[eval(f3),eval(f4), lam] if i==dump_line
                s+=1
            end
        break
        end
    end
    arr.append(s)
```

PHRUG
Philippine Ruby Users Group

Step 11

Add final conditional to get code passing

```ruby
1000.times do |i|
    p arr[-1]
    s=1
    # print arr[-40..] if i==dump_line
    (0..i).each do |f|
        s+=1 if [216,436,657,877].include?(i)
        gijs_dump[eval(f1),eval(f2), lam] if i==dump_line
        if lam[eval(f1),eval(f2)]
            s+=1
            gijs_dump[eval(f2),eval(f3), lam] if i==dump_line
            if lam[eval(f2),eval(f3)]
                s+=1
                gijs_dump[eval(f3),eval(f4), lam] if i==dump_line
                if lam[eval(f3),eval(f4)]
                    s+=1
                end
            break
            end
        break
```

PHRUG
Philippine Ruby Users Group

Step 12

Refactor eval strings into a single lambda taking 1 arg

```
fa =->(k) {"arr[-#{k}-#{k}*f..-#{k}-#{k-1}*f]"}

1000.times do |i|
    p arr[-1]
    s=1
    # print arr[-40..] if i==dump_line
    (0..i).each do |f|
        s+=1 if [216,436,657,877].include?(i)
        gijs_dump[eval(fa[1]),eval(fa[2]), lam] if i==dump_line
        if lam[eval(fa[1]),eval(fa[2])]
            s+=1
            gijs_dump[eval(fa[2]),eval(fa[3]), lam] if i==dump_line
            if lam[eval(fa[2]),eval(fa[3])]
                s+=1
                gijs_dump[eval(fa[3]),eval(fa[4]), lam] if i==dump_l
                if lam[eval(fa[3]),eval(fa[4])]
                    s+=1
                end
```

PHRUG
Philippine Ruby Users Group

Step 13

Use lambda recursively

Can’t replace last one because of syntax error when trying to put ‘break’ inside lambda

```
arr=[1]
lam = -> (p1, p2) { p1 == p2 }

fa =->(k) {"arr[-#{k}-#{k}*f..-#{k}-#{k-1}*f]"}
r = ->(n) {"if lam[eval(fa[#{n}]),eval(fa[#{n+1}])];s+=1;eval(r[#{n+1}]);end"}
1000.times do |i|
    p arr[-1]
    s=1
    (0..i).each do |f|
        s+=1 if [216,436,657,877].include?(i)
        if lam[eval(fa[1]),eval(fa[2])]
            s+=1
            eval(r[2]);break
        end
    end
    arr.append(s)
end
```

Pass (2614ms)

Expected

```
1
1
2
1
1
2
2
2
```

Output

```
1
1
2
1
1
2
2
2
```

PHRUG
Philippine Ruby Users Group

Step 14

Minify everything

```
arr=[1]
l=->(p,q){p==q}
h=->(k){"arr[-#{k}-#{k}*f..-#{k}-#{k-1}*f]"}
r=->(n){"if l[eval(h[#{n}]),eval(h[#{n+1}])];s+=1;eval(r[#{n+1}]);end"}
1000.times do|i|
p arr[-1]
s=1
(0..i).each do|f|
s+=1 if [216,436,657,877].include?(i)
if l[eval(h[1]),eval(h[2])]
s+=1
eval(r[2]);break
end
end
arr.append(s)
end
```

Pass (2614ms)

Expected

```
1
1
2
1
1
2
2
2
```

Output

```
1
1
2
1
1
2
2
2
```

PHRUG
Philippine Ruby Users Group

Before

```
g=[1]
f1 = "g[-1-f..]"
f2 = "g[-2-2*f..-2-f]"
f3 = "g[-3-3*f..-3-2*f]"
f4 = "g[-4-4*f..-4-3*f]"
1000.times do |i|
p g[-1]
            s=1
            (0..i).each do |f|
            if [216,436,657,877].include?(i);s+=1;
# print "f: #{f}, f1: #{eval(f1)}, f2: #{eval(f2)} #{eval(f1) == eval(f2)}, f3: #{eval(f3)}
#{eval(f2) == eval(f3)}, f4: #{eval(f4)} #{eval(f3) == eval(f4)} "
            end
            if eval(f1) == eval(f2)
            s+=1
            if eval(f2) == eval(f3)
            s+=1
            if eval(f3) == eval(f4)
            s+=1
            end
            break
            end
            break
            end
            end
            g.append(s)
end
```

After

```
arr=[1]
l=->(p,q){p==q}
h=->(k){"arr[-#{k}-#{k}*f..-#{k}-#{k-1}*f]"}
r=->(n){"if l[eval(h[#{n}]),eval(h[#{n+1}])];s+=1;eval(r[#{n+1}]);end"}
1000.times do|i|
p arr[-1]
s=1
(0..i).each do|f|
s+=1 if [216,436,657,877].include?(i)
if l[eval(h[1]),eval(h[2])]
s+=1
eval(r[2]);break
end
end
arr.append(s)
end
```

PHRUG
Philippine Ruby Users Group